目　录

免责申明

本文所提及的内容关系到您的安全以及合法权益与责任。使用本产品之前，请仔细阅读本文以确保已对产品进行正确的设置。不遵循和不按照本文的说明与警告来操作可能会给您和周围的人带来伤害，损坏本产品或其它周围的物品。本文档及本产品所有相关的文档最终解释权归爱华芯（大连）科技有限公司所有。如有更新，恕不另行通知。请访问爱华芯官网以获取最新的产品信息。
一旦使用本产品，即视为您已经仔细阅读本免责声明与警告、理解、认可和接受本声明全部条款和内容。您承诺对使用本产品以及可能带来的后果负全部责任。您承诺仅出于正当目的使用本产品，并且同意本条款以及爱华芯科技制定的任何相关条例、政策和指引。爱华芯科技对于直接或间接使用本产品而造成的损坏、伤害以及任何法律责任不予负责。用户应遵循包括但不限于本文提及的所有安全指引。
即使存在上述规定，消费者权益依然受当地法律法规所保障，并不受本免责声明影响。

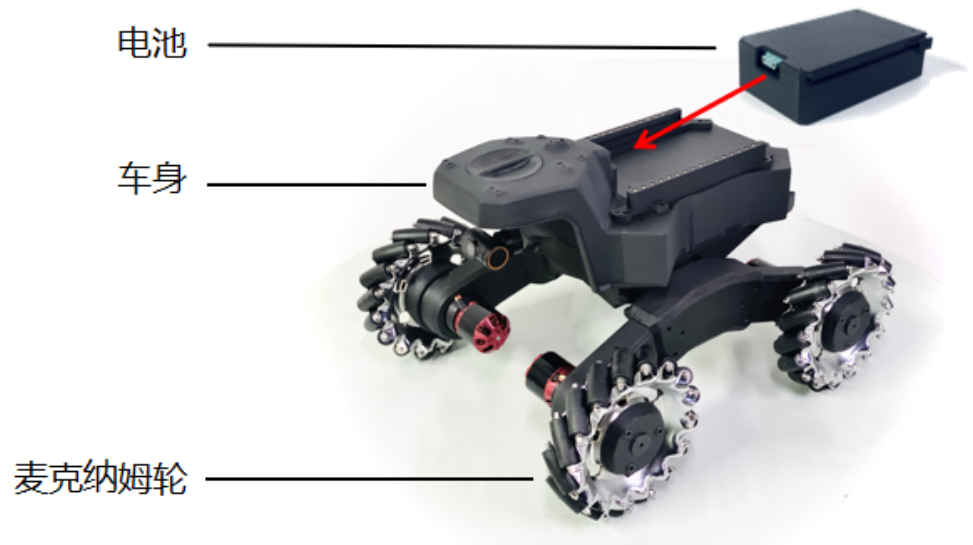
警告！

1. 务必使用爱华芯指定连接线，并严格按照各接口定义连接外部设备。
2. 严禁擅自拆解i4WD无人车与iM4WD无人车系列产品及其配件。
3. 防止水、油、沙等进入机身内部。
4. 选择合适的位置进行安装，确保散热良好。
5. 部件工作时会发热，请勿用手直接接触，否则可能造成烫伤。
6. 使用、储存及运输时，避免震动和撞击。
7. 连接至i4WD无人车或iM4WD无人车的Typec 3.0 设备可能会对 GNSS、Wi-Fi 等信号产生干扰，必要时可采取电磁屏蔽措施以减小干扰。

实验一、麦克纳姆轮车

1.1 配件介绍

实验机器人是一款名为iM4WD的全向全驱麦克纳姆轮无人车。



该车的主要配件和性能参数如下。

1.2 麦克纳姆轮车原理

1.2.1 实验目标

- 了解麦克纳姆轮的运动原理
- 了解麦克纳姆轮能够实现的运动效果

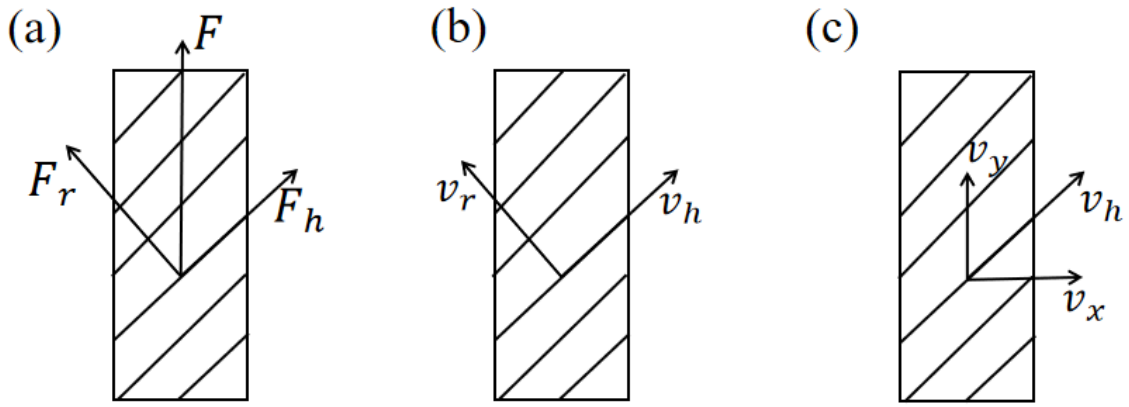
1.2.2 实验内容

图1.2中所示的是本实验搭建的全向移动机器人所使用的全向轮，麦克纳姆轮（Mecanum）。它是由瑞典Mecanum AB公司的工程师Bengt Ilon在1973年所设计开发的，因此也被称作瑞典轮或Ilon轮，自问世以来便受到工业领域的热衷。麦克纳姆轮的典型结构是轮毂表面排布一定数量的辊子，其中辊子通常是外形为椭圆弧的圆柱面，多为橡胶材质，可自由滚动，排列方式为与轮毂轴呈45°夹角。

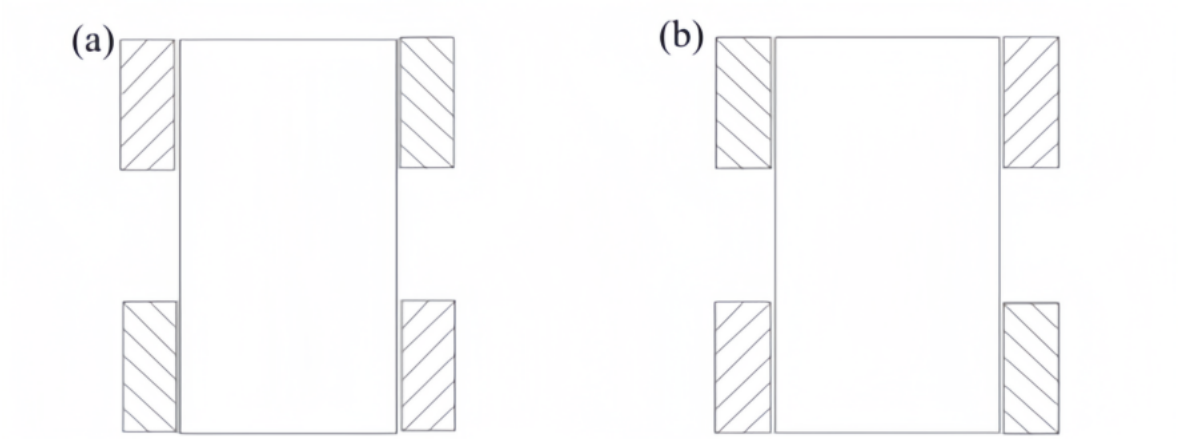
本实验机器人选用的麦克纳姆轮是WHEELTEC公司抱紧式联轴器麦克纳姆轮，规格参数如下：

名称	参数	名称	参数
辊子个数	16个	轮宽	45mm
辊子材质	橡胶	直径	152mm
辊子排列方式	45°	单轮重量	2.85kg
辊子直径	3mm	单轮承重	25kg
轮毂材质	不锈钢（表面电镀）	安装孔径	12mm

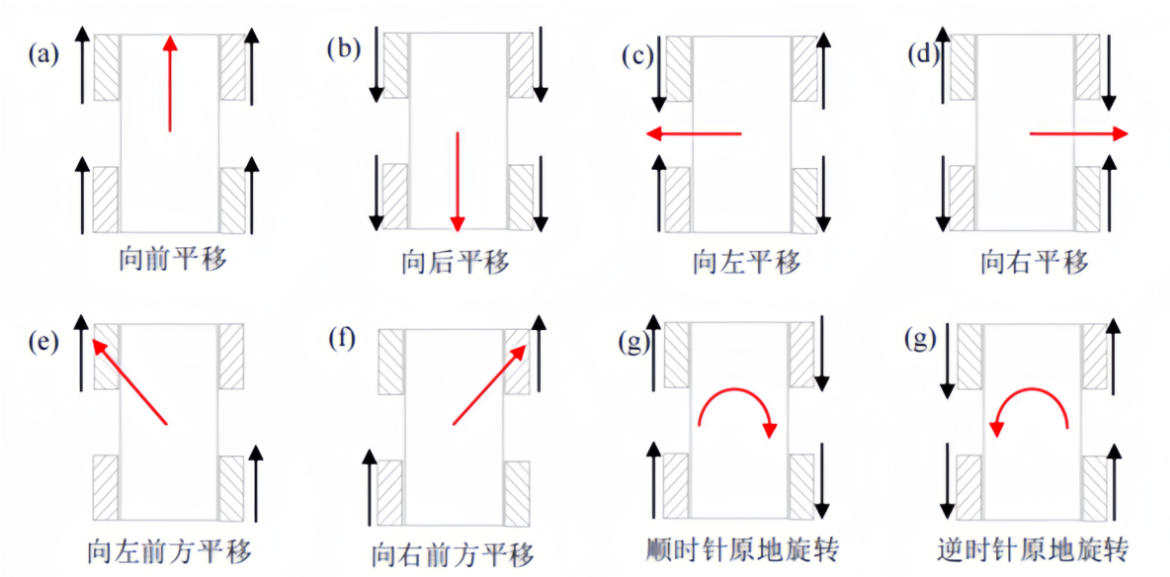
在典型的麦克纳姆轮中，辊子通常以与轮毂轴线呈45°夹角的方式安装在轮毂面上。这种特殊的结构也造就了麦克纳姆轮独特的受力分解方式，也是基于麦克纳姆轮的移动机器人能够实现全向移动的结构基础。



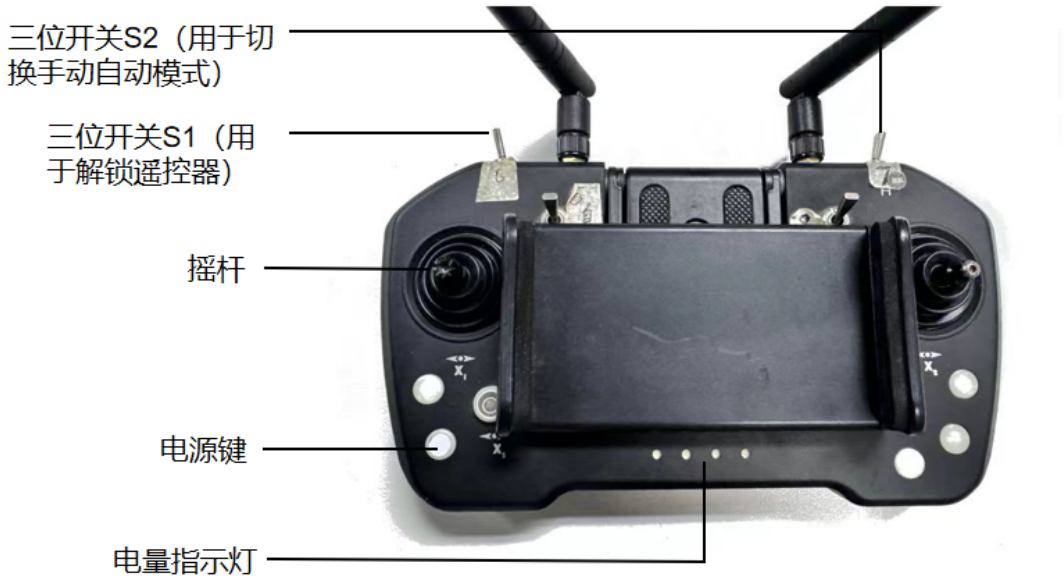
麦克纳姆轮的受力分解如图1.3(a)所示，其中矩形表示轮毂，斜线表示安装在轮毂表面的辊子。在机器人运动过程中，辊子与地面直接接触，接触点受力作用，由于辊子的特性，作用力被分解为垂直于辊子轴线的力和平行于辊子轴线的力。垂直于辊子的产生使辊子转动的速度，平行于辊子轴线的力产生驱动轮毂转动的速度，如图1.3(b)所示。由于主要驱动辊子发生转动，对移动机器人的实际速度的作用微乎其微，因此移动机器人的实际运动速度主要是由产生的。因为辊子以与轮毂轴线 45°夹角的方式排布在轮毂面上，所以可以对速度进行正交分解，如图1.3(c)所示，令垂直于轮毂轴线的速度分量为，平行于轮毂轴线的速度分量为。由麦克纳姆轮的运动特性可知，多个麦克纳姆轮通过不同的布局方式，可以得到不同的运动效果，通常，基于四个麦克纳姆轮的全向移动机器人有两种排布方式，“X”型排布和“O”型排布，如图1.4所示。



本实验搭建的平台选择“X”型排列进行布局。由前面的对麦克纳姆轮的运动原理的分析可得，基于麦克纳姆轮的全向移动机器人的运动速度是由车身上轮子的速度通过矢量合成而来的，因此通过驱动电机速度不同的转速，可以使运动底盘实现不同的运动效果，理论上可以实现的基本的八种运动效果如图1.5所示，其中黑色箭头表示轮子的运动方向，红色箭头表示整车运动方向。



接下来进行机器人遥控实操。本实验机器人使用的遥控器如图1.6所示。



首先启动遥控器，先短按后长按电源开关至遥控器的指示灯全部亮起，遥控器开启时有蜂鸣器提示音。遥控器启动后，拨动遥控器左上方的拨杆至最右端解锁遥控器，遥控器右上方拨杆保持在最右端的手动模式，就可以手动遥控机器人进行全向运动。遥控器摇杆使用方法如下：

摇杆	功能
左摇杆向上推动	机器人前进
左摇杆向下推动	机器人后退
左摇杆向左推动	机器人向左横移
左摇杆向右推动	机器人向右横移

摇杆	功能
右摇杆向左推动	机器人左转
右摇杆向右推动	机器人右转

机器人遥控演示效果如下：

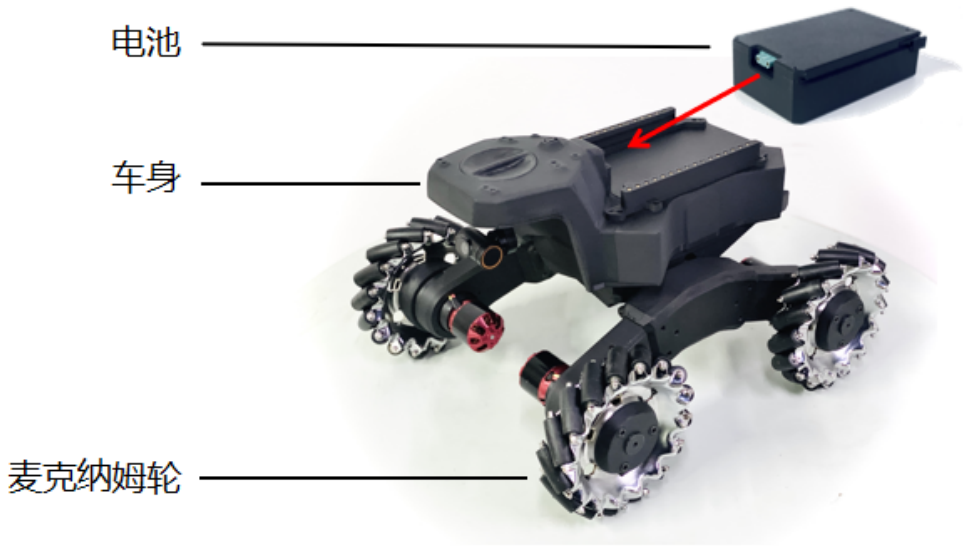
- 前进后退

- 左右横移

- 左转右转

1.1 配件介绍

实验机器人是一款名为iM4WD的全向全驱麦克纳姆轮无人车。



该车的主要配件和性能参数如下。

项目	参数
产品型号	iM4WD
尺寸重量	长43cm宽35cm高28cm；7.9kg（不含电池）
电池参数	1.510g，14.8V，20000mAh，支持5V,9V,12V,15V,20V共100W功率输出。
转向方式	差速转向，全向全驱。
产品续航	单电池2.5小时，上下双电池5个小时。
最大速度	15km/h
自主导航	挂载定位系统可实现地面站规划及自动返航，实时引导路径规划导航。
二次开发	支持ROS 开发及地面站APP开发。
地面站	支持安卓手机/平板地面站，监控无人车状态及图像回传等。
图像传输	480p实时图像传输，可拍照录像。
遥控距离	室内2.4G遥控无遮挡距离300m以上/手柄遥控距离30米左右。

项目	参数
最大载重	25kg,可根据需求定制100kg以上。
定制扩展	双排各16个10mm间距M3安装孔，方便挂载各种负载；支持顶部安装负载。支持各类485/232云台、热成像仪、夜视仪，立体相机、激光雷达、毫米波雷达、超声测量仪，喊话器、探照灯、机械手等设备。
行业应用	应用于室内无人巡检、无人安防、科研、勘探、物流等领域。

1.2 麦克纳姆轮车原理

目标

- **了解麦克纳姆轮的运动原理**
- **了解麦克纳姆轮车能够实现的运动效果**

内容

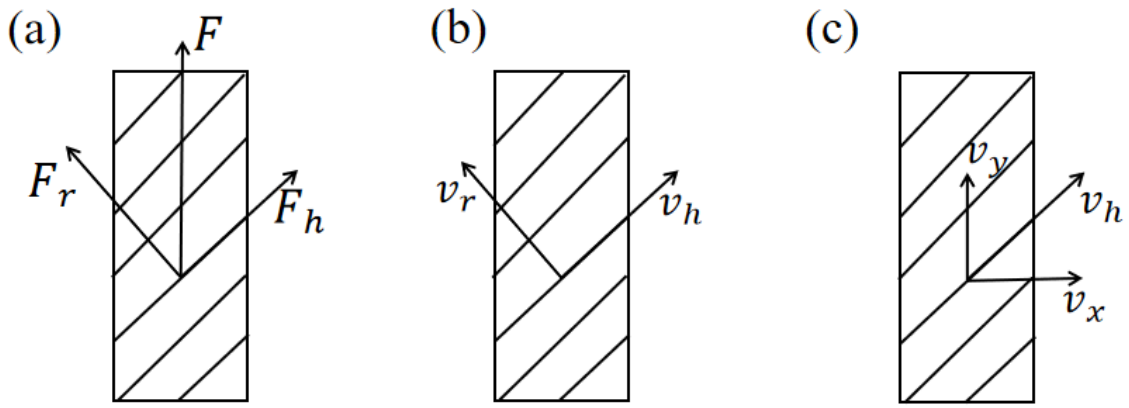
图1.2中所示的是本实验搭建的全向移动机器人所使用的全向轮，麦克纳姆轮（Mecanum）。它是由瑞典Mecanum AB公司的工程师Bengt Ilon在1973年所设计开发的，因此也被称作瑞典轮或Ilon轮，自问世以来便受到工业领域的热衷。麦克纳姆轮的典型结构是轮毂表面排布一定数量的辊子，其中辊子通常是外形为椭圆弧的圆柱面，多为橡胶材质，可自由滚动，排列方式为与轮毂轴呈45°夹角。

本实验机器人选用的麦克纳姆轮是WHEELTEC公司抱紧式联轴器麦克纳姆轮，规格参数如下：

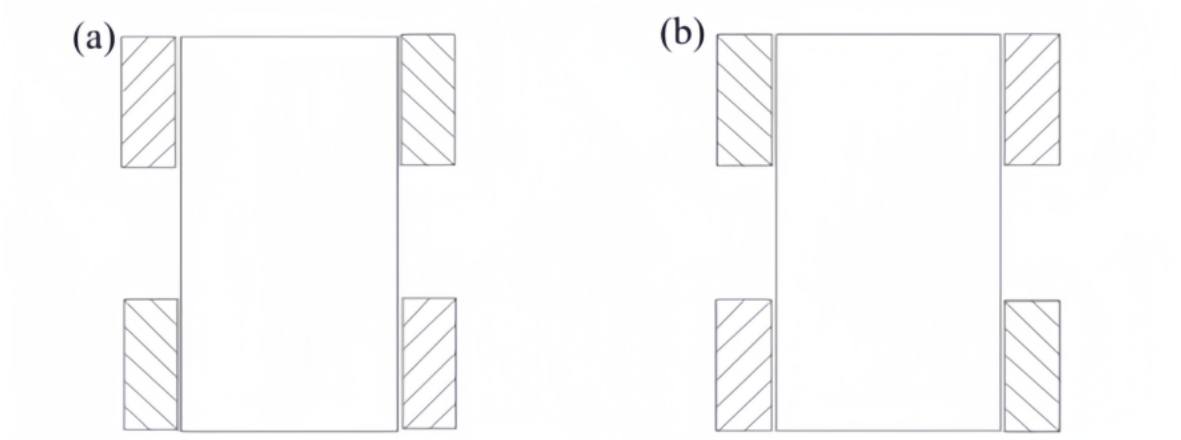
名称	参数	名称	参数
辊子个数	16个	轮宽	45mm
辊子材质	橡胶	直径	152mm
辊子排列方式	45°	单轮重量	2.85kg

名称	参数	名称	参数
辊子直径	3mm	单轮承重	25kg
轮毂材质	不锈钢（表面电镀）	安装孔径	12mm

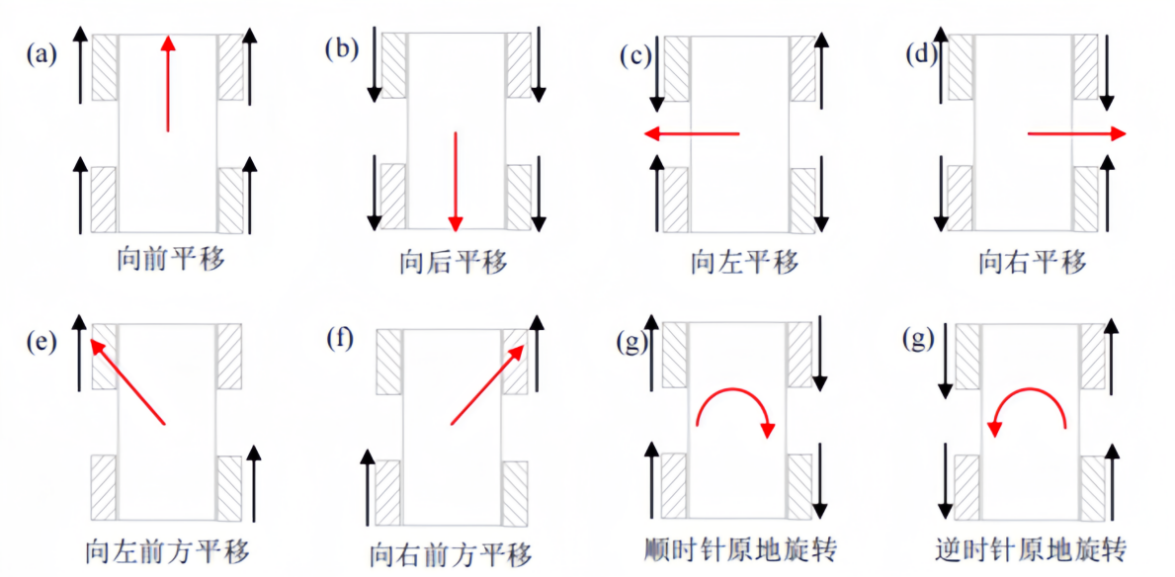
在典型的麦克纳姆轮中，辊子通常以与轮毂轴线呈45°夹角的方式安装在轮毂面上。这种特殊的结构也造就了麦克纳姆轮独特的受力分解方式，也是基于麦克纳姆轮的移动机器人能够实现全向移动的结构基础。



麦克纳姆轮的受力分解如图1.3(a)所示，其中矩形表示轮毂，斜线表示安装在轮毂表面的辊子。在机器人运动过程中，辊子与地面直接接触，接触点受力作用，由于辊子的特性，作用力被分解为垂直于辊子轴线的力和平行于辊子轴线的力。垂直于辊子的产生使辊子转动的速度，平行于辊子轴线的力产生驱动轮毂转动的速度，如图1.3(b)所示。由于主要驱动辊子发生转动，对移动机器人的实际速度的作用微乎其微，因此移动机器人的实际运动速度主要是由产生的。因为辊子以与轮毂轴线 45°夹角的方式排布在轮毂面上，所以可以对速度进行正交分解，如图1.3(c)所示，令垂直于轮毂轴线的速度分量为，平行于轮毂轴线的速度分量为。由麦克纳姆轮的运动特性可知，多个麦克纳姆轮通过不同的布局方式，可以得到不同的运动效果，通常，基于四个麦克纳姆轮的全向移动机器人有两种排布方式，“X”型排布和“O”型排布，如图1.4所示。



本实验搭建的平台选择“X”型排列进行布局。由前面的对麦克纳姆轮的运动原理的分析可得，基于麦克纳姆轮的全向移动机器人的运动速度是由车身上轮子的速度通过矢量合成而来的，因此通过驱动电机速度不同的转速，可以使运动底盘实现不同的运动效果，理论上可以实现的基本的八种运动效果如图1.5所示，其中黑色箭头表示轮子的运动方向，红色箭头表示整车运动方向。

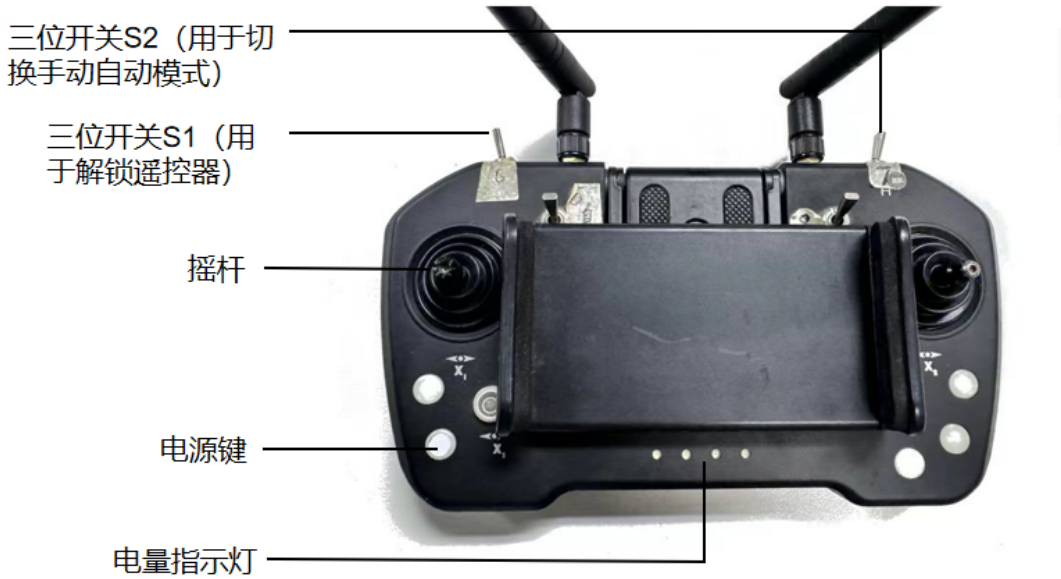


机器人遥控实操

本实验机器人可以使用遥控器和手柄两种方式遥控。

1、遥控器遥控

本实验采用的遥控器及相关的按键介绍如下图所示：



使用遥控器控制机器人运动时，首先启动遥控器，先短按后长按电源开关至遥控器的指示灯全部亮起，遥控器开启时有蜂鸣器提示音。遥控器启动后，拨动遥控器左上方的拨杆至最右端解锁遥控器，遥控器右上方拨杆保持在最右端的手动模式，就可以手动遥控机器人进行全向运动。遥控器摇杆使用方法如下：

摇杆	功能
左摇杆向上推动	机器人前进
左摇杆向下推动	机器人后退

摇杆	功能
左摇杆向左推动	机器人向左横移
左摇杆向右推动	机器人向右横移
右摇杆向左推动	机器人左转
右摇杆向右推动	机器人右转

机器人遥控演示效果如下：

- 前进后退

- 左右横移

- 左转右转

2、手柄遥控

本实验采用ROS专用的USB遥控手柄，操作更加灵活，并且可以直接远程扩展机器人。手柄的相关按键介绍及参数如下：

项目	参数
可自定义功能键数量	23个
尺寸（长宽高）	15.5cm,9.5cm,5.5cm
摇杆输出模式	支持输出模拟值
支持系统	支持Windows、Linux
通讯方式	USB-A接收段子+2.4G无线通讯
接口类型	USB2.0免驱
通讯距离	约15m（与使用环境有关）

手柄摇杆使用方法如下：

摇杆	功能
左摇杆向上推动	机器人前进
左摇杆向下推动	机器人后退
左摇杆向左推动	机器人向左横移
左摇杆向右推动	机器人向右横移
右摇杆向左推动	机器人左转

摇杆	功能
右摇杆向右推动	机器人右转

1.3 实验小结

- **问题思考**

  在进行遥控操作时如何控制机器人的运动速度？

- **注意事项**

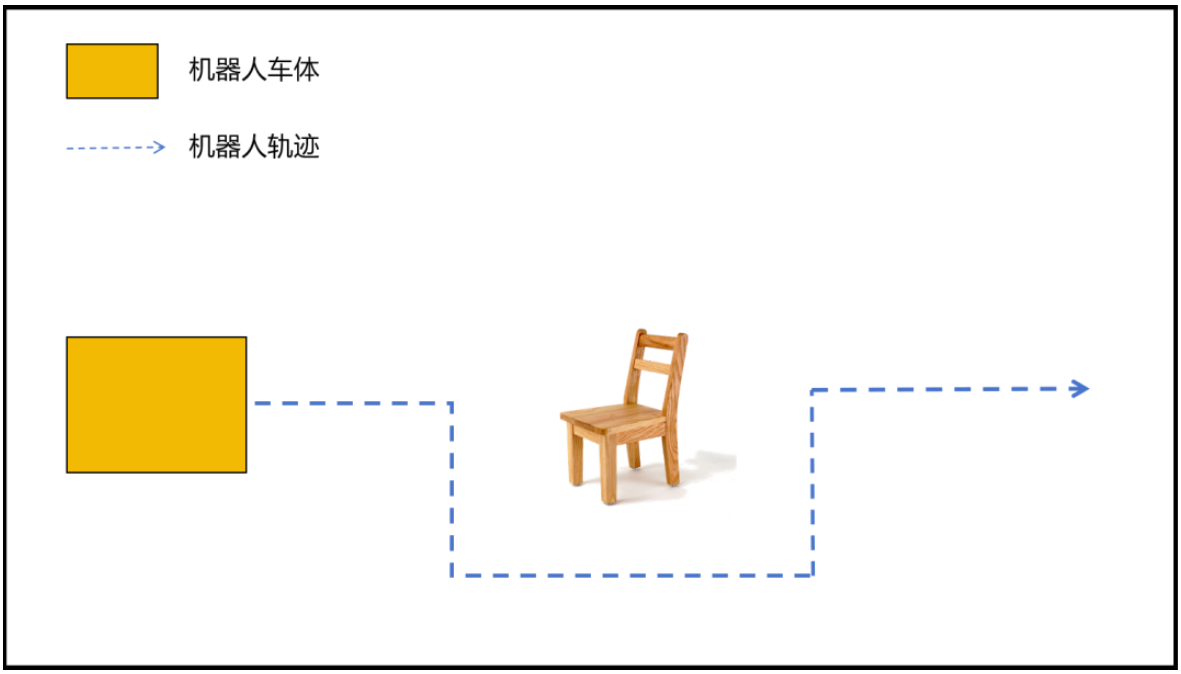
  在进行转向等运动时尽量缓慢操作，保持平稳。

  操作机器人运动时速度不宜太快，容易损伤车轮。

  实验结束后先锁定遥控器再关机。

- **随堂作业**

  在实验室场地中，摆放一个椅子(或场地中其他可用物品）作为障碍物，独立使用遥控器操作机器人运动，使机器人在前进过程中能够绕过障碍物并继续保持直行。



实验二、Ardupilot控制系统

2.1 MP(MissionPlanner)

MP(MissionPlanner)简介

Misson Planner简称MP，是Windows 平台运行的一款APM/PIX的专属地面站。它也是一款完全开放源码的地面站。Mission Planner作为Ardupilot控制系统的强大工具，它为用户提供了一个直观易用的界面，用于配置、监控和控制飞行器。通过Mission Planner，用户可以轻松地管理Ardupilot飞行器，并实现各种复杂的任务。
官网链接：Ardupilot官网。
源码链接：GitHub源码

它主要功能包括：

- 对APM/PIX进行烧录固件
- 安装、配置和优化参数
- 规划航点任务，可以使用google地图或者其它地图
- 下载和分析飞行日志
- 使用专用的PC飞行模拟软件接口，进行硬件模拟飞行
- 连接一个遥测数传，你还可以：
    - 实时监控飞行器状态
    - 记录一个实时遥测日志
    - 查看和分析遥测日志
    - 在FPV中操作无人机（第一人称视角

主要窗口功能包括：

1. 软件版本号，连接飞控后，会同时显示飞控的软件版本号
2. 主要功能选项区域
3. 连接方式选择和连接断开控制
4. 飞机仪表盘，包括姿态、高度、GPS状态、飞行模式
5. 状态和其他辅助控制功能区域
6. GPS地图，连接上飞控和GPS后，会显示一个实时位置和飞行轨迹
7. GPS坐标和精度显示


已锁定
EKF
Vibe
GPS: 无GPS
快速
动作
起飞前检查
仪表
状态
舵机
遥测日志
数据闪存日志
脚本
消息
高度 (m)
地速 (m/s)
航点距离 (m)
偏航 (deg)
升降速度 (m/s)
DistToMAV
115200
调试
自动平移
缩放

2.2 Mavros定位

目标

- **了解如何使用激光里程计和Mavros在ROS2中进行定位**

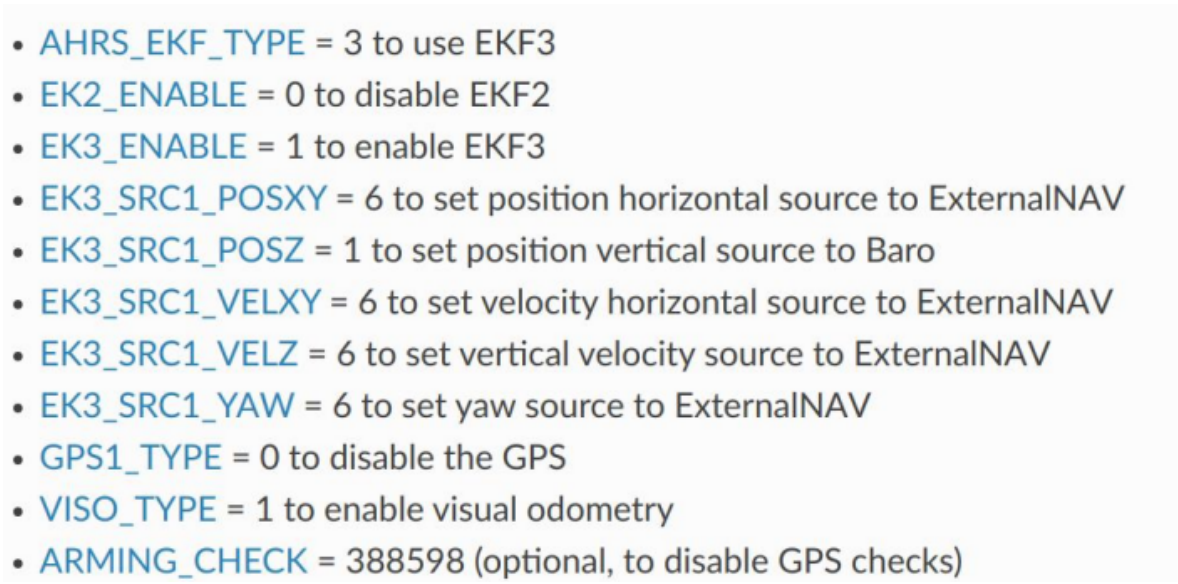
内容

1、搭建环境

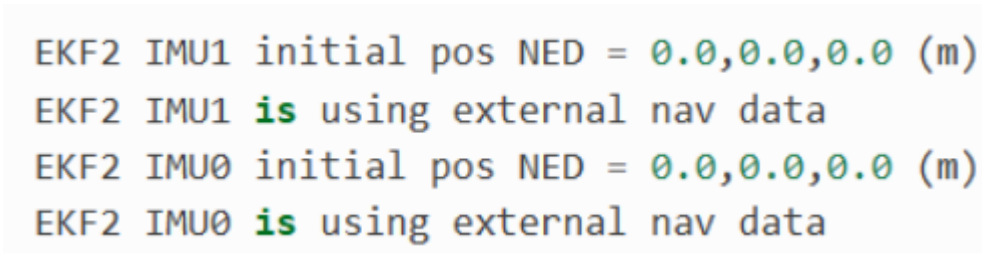
- 安装ROS2。
- 安装Mavros和Cartographer包。
- 准备激光雷达传感器和飞控设备，确保其可以与电脑进行通信。

2、参数设置

修改Cartographer配置文件，设置激光里程计相关参数。设置AHRS_GPS_USE=0关闭GPS使用。



3、启动SLAM
将SLAM的里程计输出话题重映射为/mavros/vision_pose/pose，然后开启mavros后可以观察到以下信息输出，表示飞控已经接收到了里程计消息。



设置EKF原点，以启动EKF算法，将里程计数据与IMU数据进行融合。通过输入命令

```
ros2 topic pub
```

发送虚假GPS消息，或者通过Mission Planner在地图中右击选择“set EKF Origin”，设置EKF原点。

输入命令

```
ros2 topic pub
```

命令发送Twist速度消息到/mavros/setpoint_velocity/cmd_vel_unstamped话题，观察机器人是否能够根据指令进行移动。

2.3 实验小结

实验三、Foxglove（机器人可视化监控界面）

3.1 实验内容

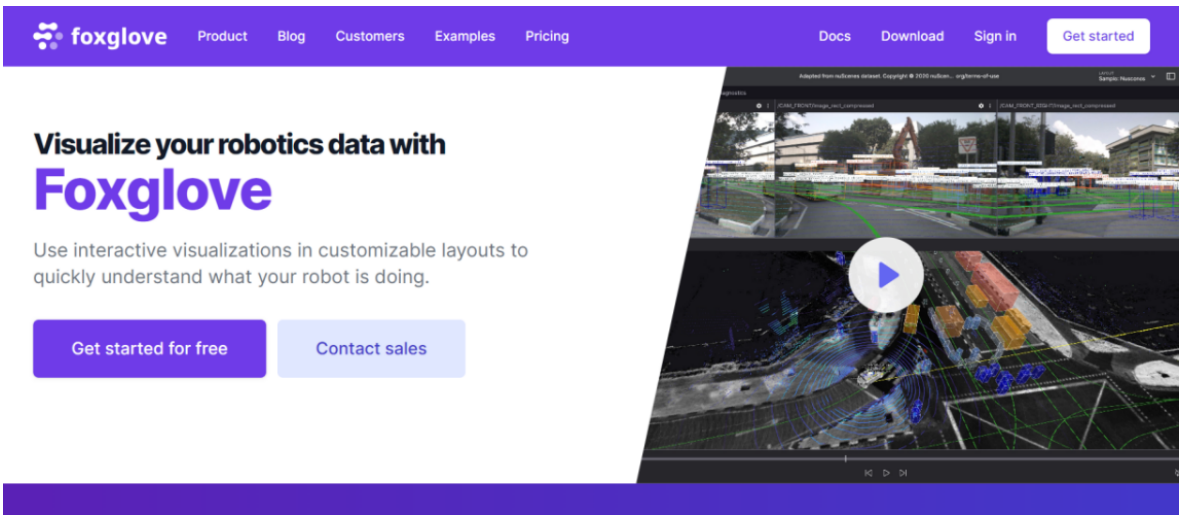
目标

- **了解Foxglove机器人可视化监控界面的优势**
- **了解Foxglove的操作流程**

内容

Foxglove简介

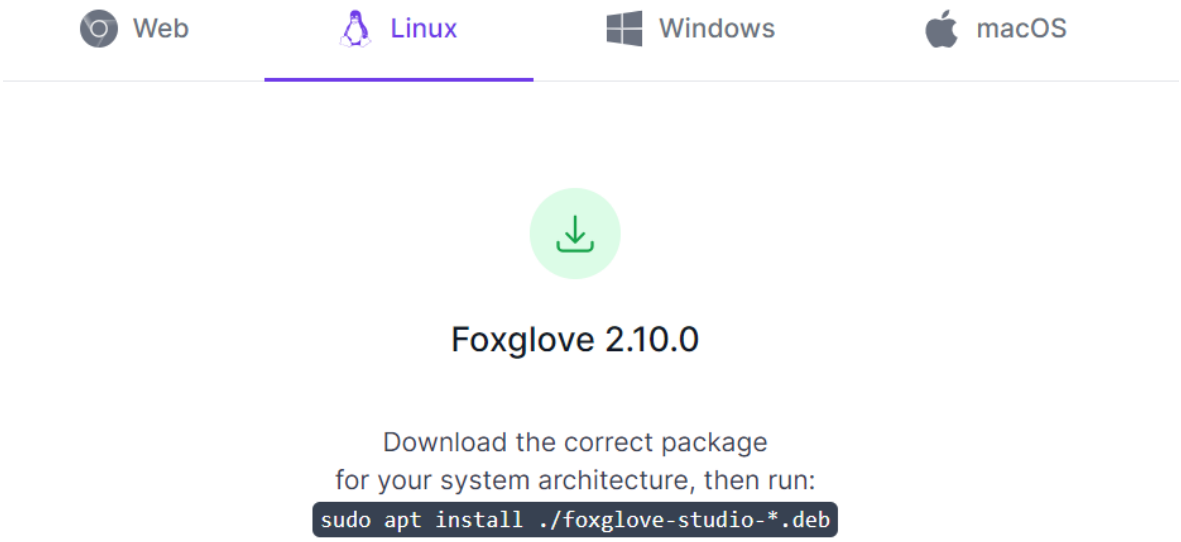
Foxglove是一个为机器人工程师和开发者设计的开源数据可视化平台，在处理复杂的机器人数据和系统监控方面具有很多优势。它允许用户查看、理解和分析机器人系统的实时数据，这对于开发和调试机器人应用程序至关重要。



通过使用Foxglove，我们可以在电脑端实时显示和监控机器人的数据，这种远程可视化的方式不仅避免了占用机器人端的内存资源，而且能够显著提高机器人系统的整体性能。这样，机器人可以更加高效地执行任务，同时确保了数据处理和监控的实时性和准确性。

Foxglove下载

首先在Foxglove下载网站下载Foxglove安装包，这里以安装Linux版本为例。



打开终端，输入：

```
sudo apt install ./foxglove-studio-*.deb
```

安装Foxglove。

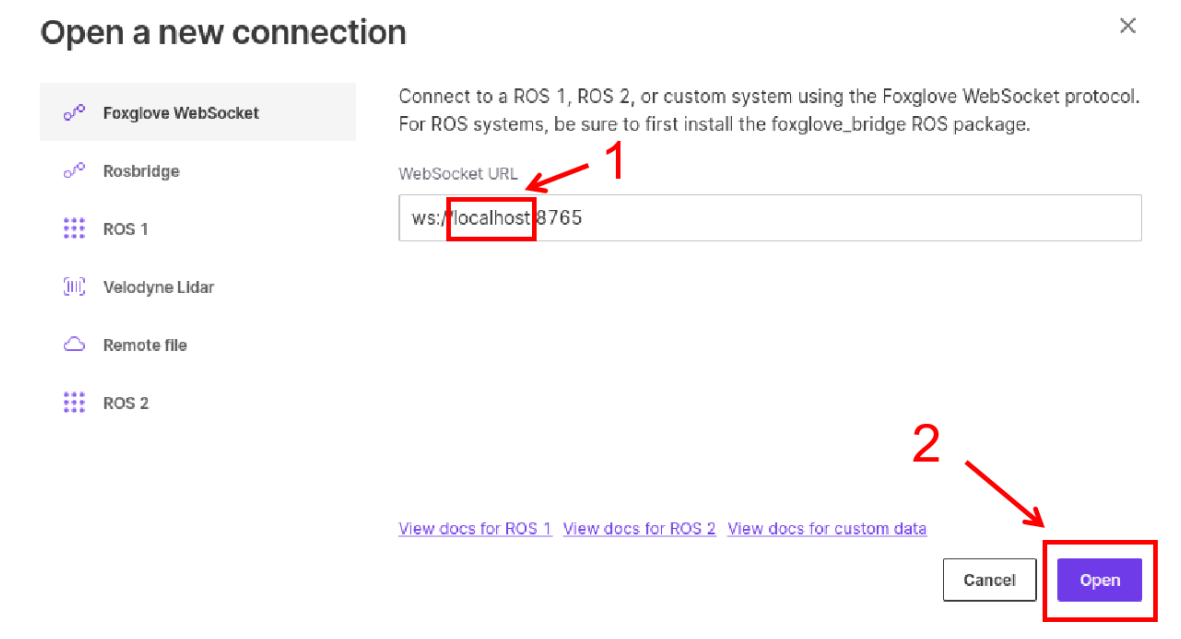
确保机器人端和电脑端处在相同局域网下，可以使用ping命令测试你的机器与服务器之间的网络连接是否正常。

首先，在机器人终端输入：

```
ros2 launch foxglove_bridge foxglove_bridge_launch.xml
```

启动foxglove_bridge的服务。

打开Foxglove。单击“打开连接”菜单，将URL中的”localhost”替换为要访问的目标IP，单击“Open”，即可建立连接。



连接成功后就可以按照实验五、实验六中的操作载入存放ros消息的包，并在Foxglove中显示可视化结果。这里以实验四中获取的点云数据为例，按照实验四中的步骤在香橙派端载入包后可以实时观看完整的点云数据，还可以根据需求添加其他面板工具。

3.2 实验小结

实验四、ROS（机器人操作系统）

4.1 Ubuntu22.04操作系统

目标

- **了解Ubuntu22.04的基本使用**

内容

1、Ubuntu 22.04的安装

- 下载Ubuntu 22.04 ISO文件：访问 Ubuntu 官方网站下载最新版本的 ISO 文件。
- 创建启动盘：使用 Rufus 或其他软件将 ISO 文件写入U盘或DVD。
- 设置BIOS/UEFI引导顺序：进入BIOS/UEFI设置，将启动盘设为第一引导设备。
- 开始安装：重启电脑，从启动盘启动Ubuntu，按照提示完成安装过程。

2、Linux常用命令

命令	功能
cd	打开文件夹
cd..	返回上一级目录
ls	列出目录内容
tab	补全命令
Ctrl+Alt+t	打开终端
Ctrl+c	中断程序执行
mkdir	创建文件夹
sudo	切换到超级用户模式以执行超级用户权限，提示输入密码

3、软件安装和更新

- 软件中心：使用软件中心搜索并安装软件。
- 终端安装：使用apt命令安装软件，例如：sudo apt install <软件名>
- 软件更新：使用apt命令更新软件，例如：sudo apt update && sudo apt upgrade。

4.2 ROS2操作系统

目标

- **了解ROS2操作系统的基本使用**

内容

1、ROS2简介

ROS2是在ROS的基础上设计开发的第二代机器人操作系统，它是一款面向机器人开发的开源软件平台。ROS 2提供了一组工具和库，可帮助开发人员轻松构建机器人应用程序。ROS 2基于DDS（Data Distribution Service）协议，允许机器人的不同组件之间进行高效的通信。ROS 2还提供了可重复性和模块化开发的支持，使得机器人开发人员能够更加灵活和高效地开发和测试机器人应用程序。

需要注意的是，ROS本身并不是一个操作系统，而是可以安装在现在已有的操作系统上（Linux、Windows、Mac）上的软件库和工具集。

2、ROS2的安装

- 选择合适的版本：ROS2有多个版本，可以根据需求选择合适的版本。
- 安装ROS2：根据官方文档进行安装，可以使用一键安装指令等方式进行安装。
- 配置环境变量：将ROS2的环境变量添加到.bashrc或文件中。

3、创建ROS2工作空间

- 创建工作空间：使用mkdir和cd命令创建一个新的工作空间目录。
- 创建功能包：使用ros2 pkg create命令创建一个新的功能包。
- 编写节点代码：使用C++或Python编写节点代码，实现特定的功能。
- 编译节点：使用colcon build命令编译工作空间中的所有节点。

4、运行ROS2节点

- 启动节点：使用ros2 run [功能包名] [节点名]启动节点。
- 查看话题信息：使用ros2 topic list令查看当前系统中的所有话题。
- 发布话题消息：使用ros2 topic pub [话题名] [消息类型] '{数据}'发布话题消息。
- 订阅话题消息:使用ros2 topic echo [话题名]订阅话题消息。

4.3 实验小结

实验五、SLAM建图

5.1 实验内容

目标

- **了解什么是SLAM**
- **实现SLAM功能**

内容

SLAM简介

SLAM（Simultaneous Localization and Mapping），同步定位与地图构建，最早在机器人领域提出，它指的是：机器人从未知环境的未知地点出发，在运动过程中通过重复观测到的环境特征定位自身位置和姿态，再根据自身位置构建周围环境的增量式地图，从而达到同时定位和地图构建的目的。

本实验的机器人选用Livox_Mid-360激光雷达系统。

激光雷达通过连续扫描环境，收集大量数据点，对收集到的数据点进行处理后，生成点云数据，并利用处理后的数据构建环境的地图。此外，为了后续进行机器人在未知环境中的导航功能，SLAM算法将实时融合激光雷达数据与地图，进行定位和建图。

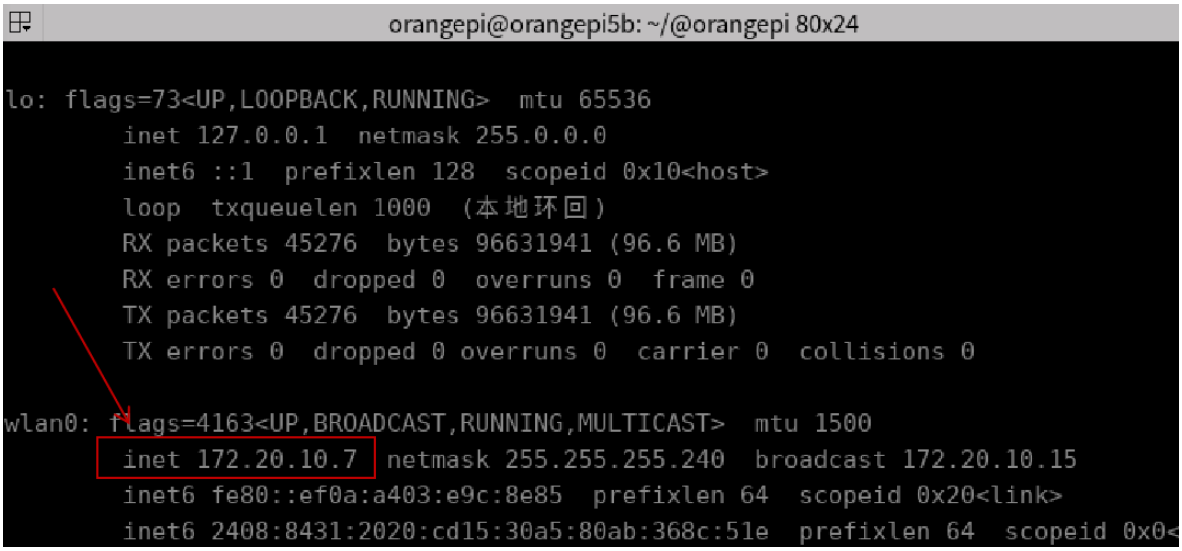
1、SSH远程连接

为了方便远程控制机器人，我们在电脑端用SSH连接机器人上位机端，这样只需要在笔记本电脑上操作，就可以直接控制机器人。

打开机器人电源和上位机电源，并确保上位机端和个人PC端连接在同一局域网下。同时按下键盘组合键“Ctrl + Alt + T”启动上位机终端，打开终端输入

```
ifconfig
```

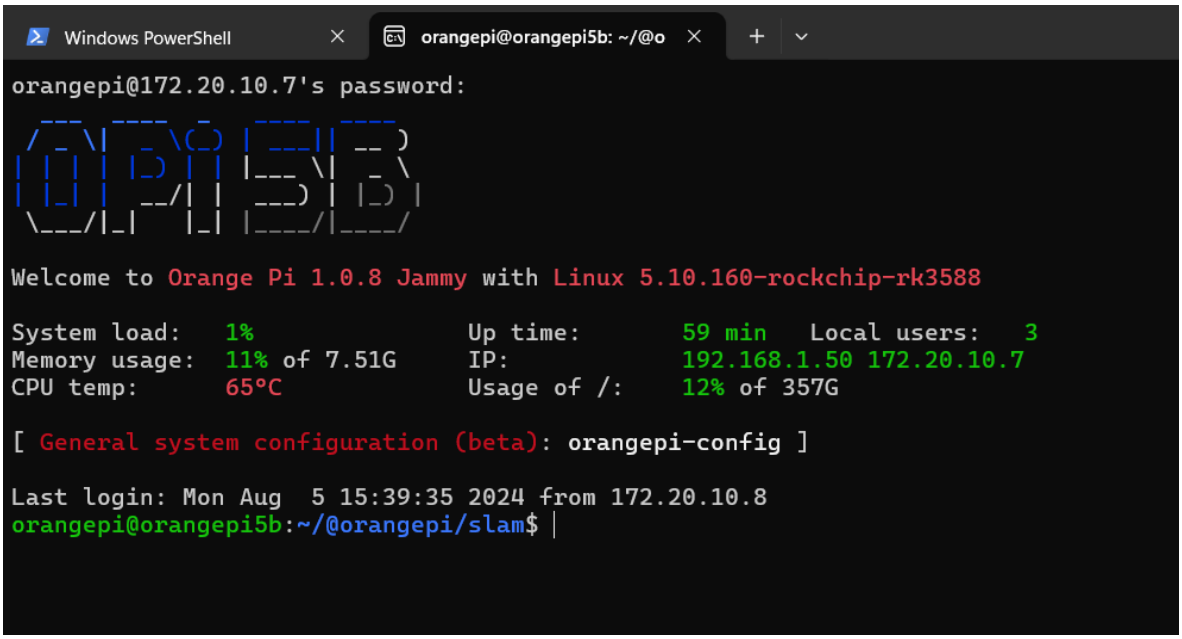
记住机器人的IP地址。



进行SSH远程连接，在电脑端启动终端，输入

```
ssh orangepi@172.20.10.7
```

输入密码“x”按回车，进入机器人终端，会发现@后面的计算机名变成了orangepi，说明当前终端已经进入了香橙派端。

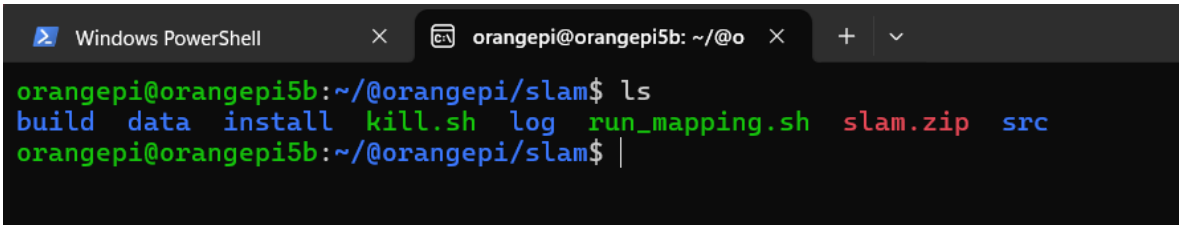


2、控制机器人运动建图

在香橙派终端打开slam文件夹，输入

```
ls
```

查看slam文件夹下的文件。



输入

> ./runmapping.sh

启动一个建图程序，检查终端窗口，检查终端中的节点是否正常启动。运行runmapping.sh脚本，在控制机器人运动的过程中收集激光雷达数据并构建环境的地图。运行SLAM算法，录制全局点云bag包。

设定机器人的起点，通过遥控器控制机器人运动。

本实验选择大黑楼走廊进行建图，通过遥控器控制机器人绕走廊运动一周作为实验设置的轨迹，通过runmapping.sh启动waypoint_save算法录制轨迹。

运动结束后，输入

> ./kill.sh

关闭SLAM算法和waypoint_save算法。

3、Rviz可视化结果

打开香橙派终端，输入

> rviz2

进入rviz可视化界面。

根据脚本设置，录制的轨迹将自动保存在data文件夹下的prior_map文件夹中。输入

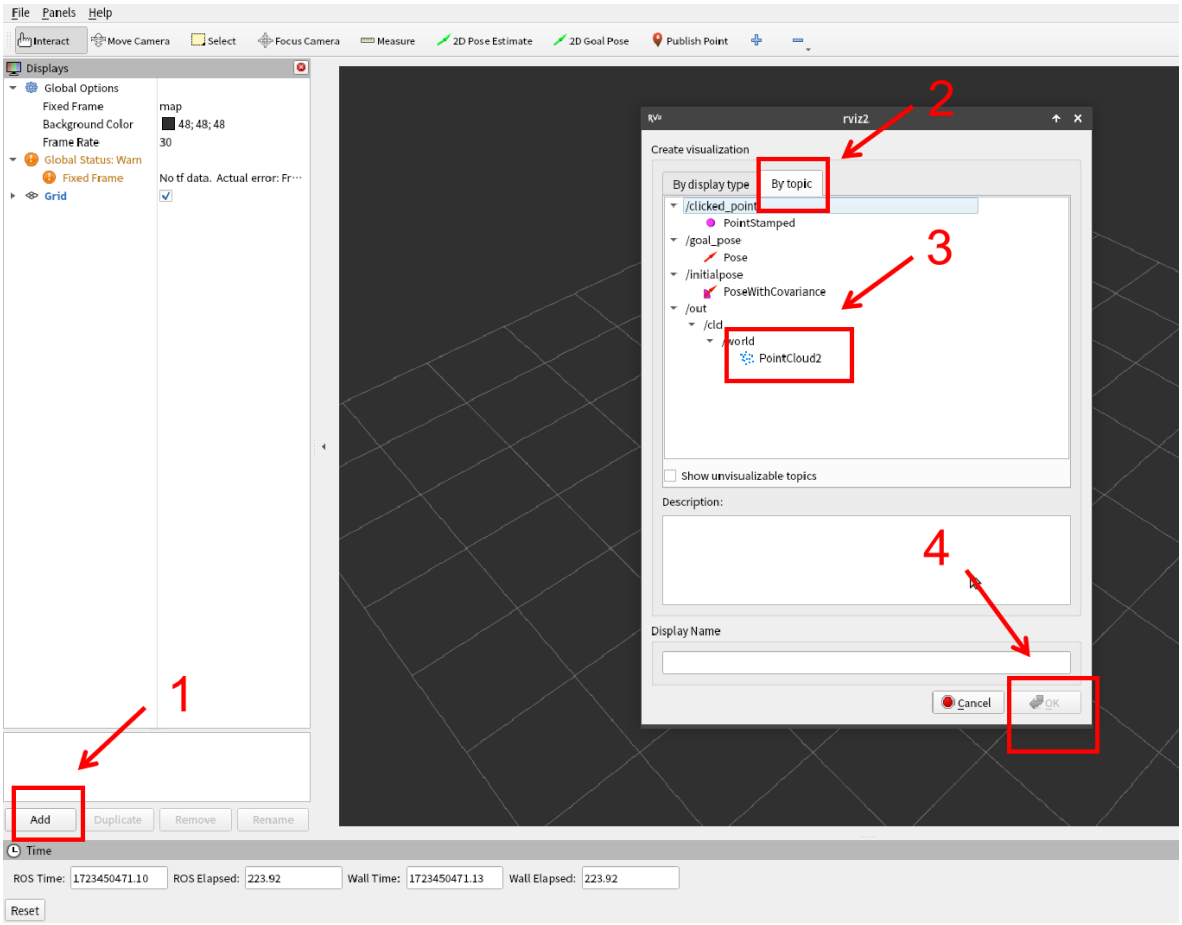
> ls

可以查看当前文件夹中的文件。这里以文件名为001的包为例。在prior_map打开终端输入

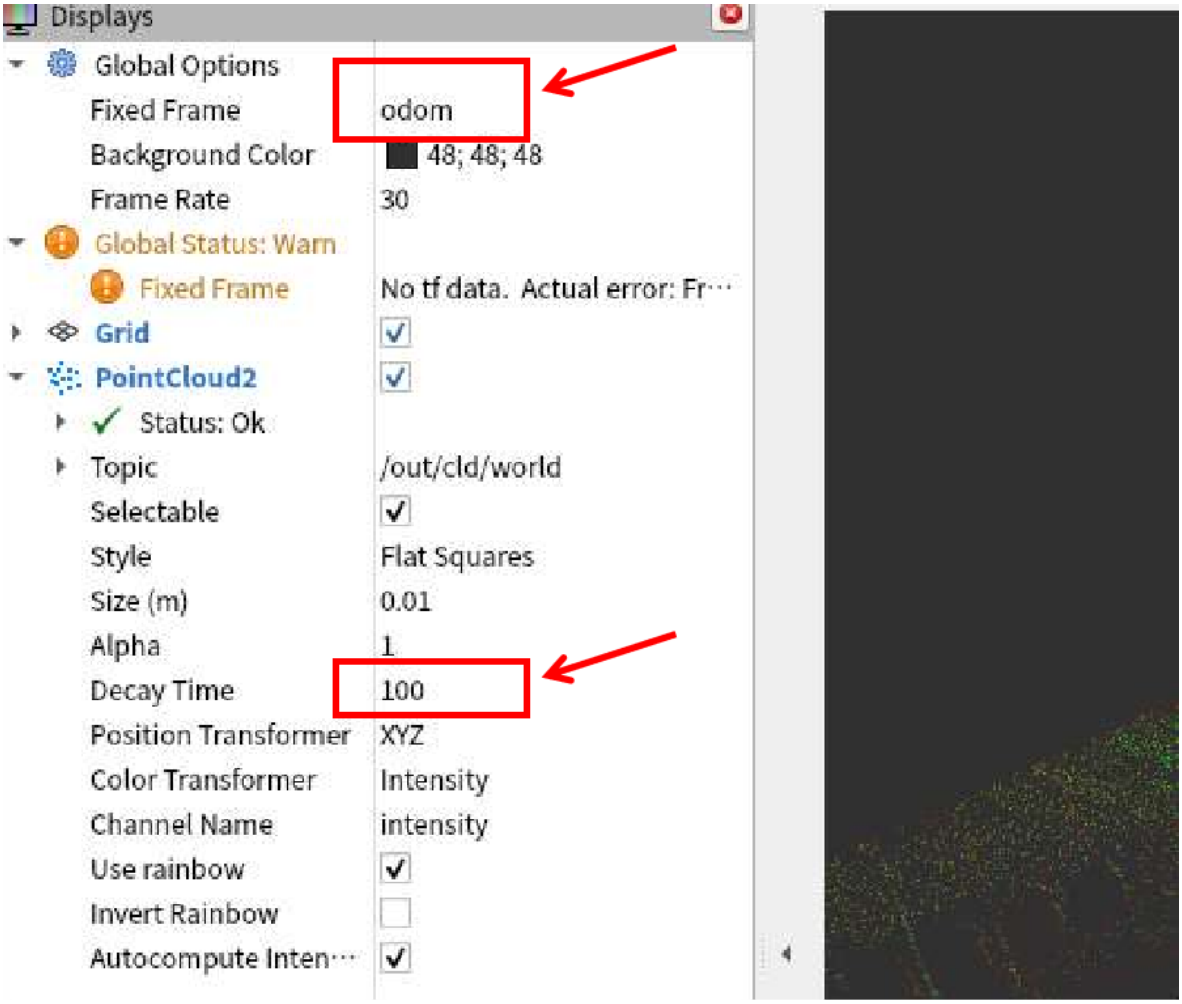
> ros2 bag play 001/

将001包中的消息载入rviz中。

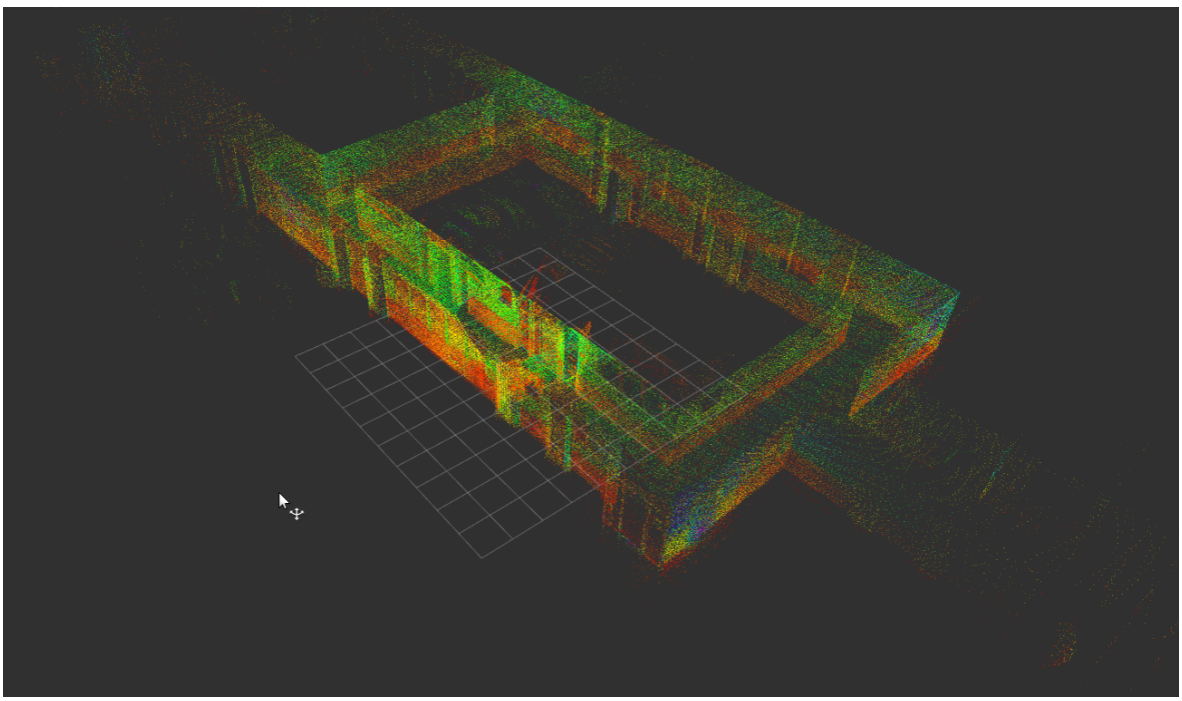
在rviz界面中，点击Add——By topic—— 选择点云数据话题 —— 点击OK。



将rviz中的Fixed_Frame参数修改为odom。设置“decay time”（衰减时间）为100，以得到更加清晰稳定的点云数据。



等待一段时间后，可以显示出建图得到的完整点云。



5.2 实验小结

实验六、自主导航

6.1 实验内容

目标

- **了解自主导航的实现方法**
- **用实验五里建好的地图，实现自主导航功能**

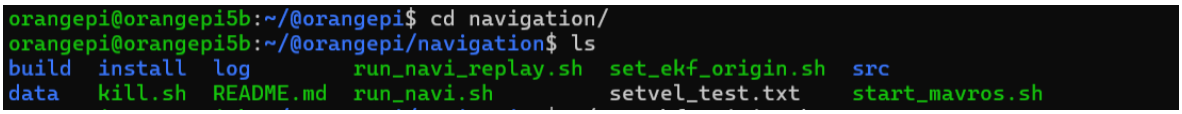
内容

1、机器人自主导航的实现

首先确保机器人已经回到实验二中设定的起点，锁定机器人遥控器。在香橙派终端打开navigation文件夹，同样可以输入

```
ls
```

查看navigation文件夹下的文件。

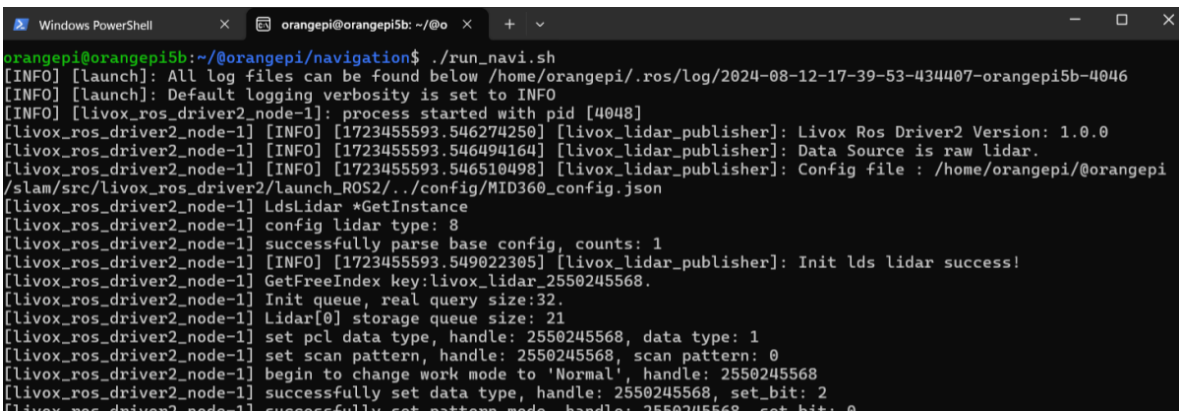


依次运行run_navi.sh、start_mavros.sh和set_ekf_origin.sh脚本，并依次检查弹终端窗口，检查每个终端中的节点是否正常启动。

输入

```
./run_navi.sh
```

启动重定位和导航算法。



输入

```
./start_mavros.sh
```

启动飞行控制单元。飞控部分负责处理传感器数据，计算机器人的姿态和位置，并生成控制信号来调整执行器的动作，从而实现对机器人的控制。

```
orangepi@orangepi5b:~/@orangepi/navigation$ ./start_mavros.sh
[INFO] [launch]: All log files can be found below /home/orangepi/.ros/log/2024-08-12-17-42-43-309182-orangepi5b-4728
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [mavros_node-1]: process started with pid [4729]
[mavros_node-1] [INFO] [1723455763.598626030] [mavros.mavros_node]: Starting mavros_node container
[mavros_node-1] [INFO] [1723455763.598765737] [mavros.mavros_node]: FCU URL: /dev/ttyS0:921600
[mavros_node-1] [INFO] [1723455763.598779445] [mavros.mavros_node]: GCS URL: udp://:14550@
[mavros_node-1] [INFO] [1723455763.598788778] [mavros.mavros_node]: UAS Prefix: /uas1
[mavros_node-1] [INFO] [1723455763.598813278] [mavros.mavros_node]: Starting mavros router node
[mavros_node-1] [INFO] [1723455763.632466014] [mavros.mavros_router]: Built-in SIMD instructions: ARM NEON
[mavros_node-1] [INFO] [1723455763.632538347] [mavros.mavros_router]: Built-in MAVLink package version: 2024.6.6
[mavros_node-1] [INFO] [1723455763.632561097] [mavros.mavros_router]: Known MAVLink dialects: common ardupilotmega ASLUA
V AVSSUAS all csAirLink cubepilot development icarous matrixpilot paparazzi standard storm32 uAvionix ualberta
[mavros_node-1] [INFO] [1723455763.632591430] [mavros.mavros_router]: MAVROS Router started
[mavros_node-1] [INFO] [1723455763.632653262] [mavros.mavros_router]: Requested to add endpoint: type: 0, url: /dev/ttyS
0:921600
[mavros_node-1] [INFO] [1723455763.632701678] [mavros.mavros_router]: Endpoint link[1000] created
[mavros_node-1] [INFO] [1723455763.633805334] [mavros.mavros_router]: link[1000] opened successfully
```

输入

```
./set_ekf_origin.sh
```

为机器人设置原点，为了与实验二的建图保持一致，选取大黑楼的地理坐标为原点。

```
orangepi@orangepi5b:~/@orangepi/navigation$ ./set_ekf_origin.sh
publisher: beginning loop
publishing #1: geographic_msgs.msg.GeoPointStamped(header=std_msgs.msg.Header(stamp=builtin_interfaces.msg.Time(sec=0, n
anosec=0), frame_id=''), position=geographic_msgs.msg.GeoPoint(latitude=38.886064, longitude=121.524577, altitude=100.0)
)

publishing #2: geographic_msgs.msg.GeoPointStamped(header=std_msgs.msg.Header(stamp=builtin_interfaces.msg.Time(sec=0, n
anosec=0), frame_id=''), position=geographic_msgs.msg.GeoPoint(latitude=38.886064, longitude=121.524577, altitude=100.0)
)

publishing #3: geographic_msgs.msg.GeoPointStamped(header=std_msgs.msg.Header(stamp=builtin_interfaces.msg.Time(sec=0, n
anosec=0), frame_id=''), position=geographic_msgs.msg.GeoPoint(latitude=38.886064, longitude=121.524577, altitude=100.0)
```

将遥控器右上方拨杆拨到左端的自动模式，机器人会按照实验二中建好的地图运动。由于纯跟踪轨迹跟踪控制算法的设定，可以选择两种运动模式。

第一，执行避障功能。进入pure_persuit文件夹下的launch文件夹，打开pure_persuit.py文件，修改避障函数is_avoid_obstacle中的参数为true，并同时修改local_planner文件夹下的local_planner.py文件，同样修改避障函数is_avoid_obstacle中的参数为true。在完成以上两处修改后，机器人在进行导航运动时将执行避障功能。执行避障功能时，当机器人前方出现障碍物，机器人将自动绕过障碍物并重新回到原路线继续行驶。

视频演示

第二，不执行避障功能。当pure_persuit.py和local_planner.py文件中函数is_avoid_obstacle的参数均设置为为false时，机器人在进行导航运动时不执行避障功能。机器人运动的过程中， 每前进五米就会暂停五秒。当机器人前方出现障碍物时，机器人会停止前进，障碍物消失后，机器人继续按照地图运动。

视频演示

当机器人返回起点，停止运动，可以在终端输入输入

```
./kill.sh
```

关闭全部进程。

2、rviz可视化结果

导航结束后，会自动生成俩个文件夹，分别保存里程计的数据和实验数据。进入navigation文件夹，进入data文件夹下的bags文件夹，输入

```
ls
```

可以查看当前文件夹下保存的包名。这里将存放里程计数据的包命名为navigation01，将存放实验数据的包命名为navigation02。

首先查看里程计数据的可视化结果。打开香橙派终端，输入

```
rviz2
```

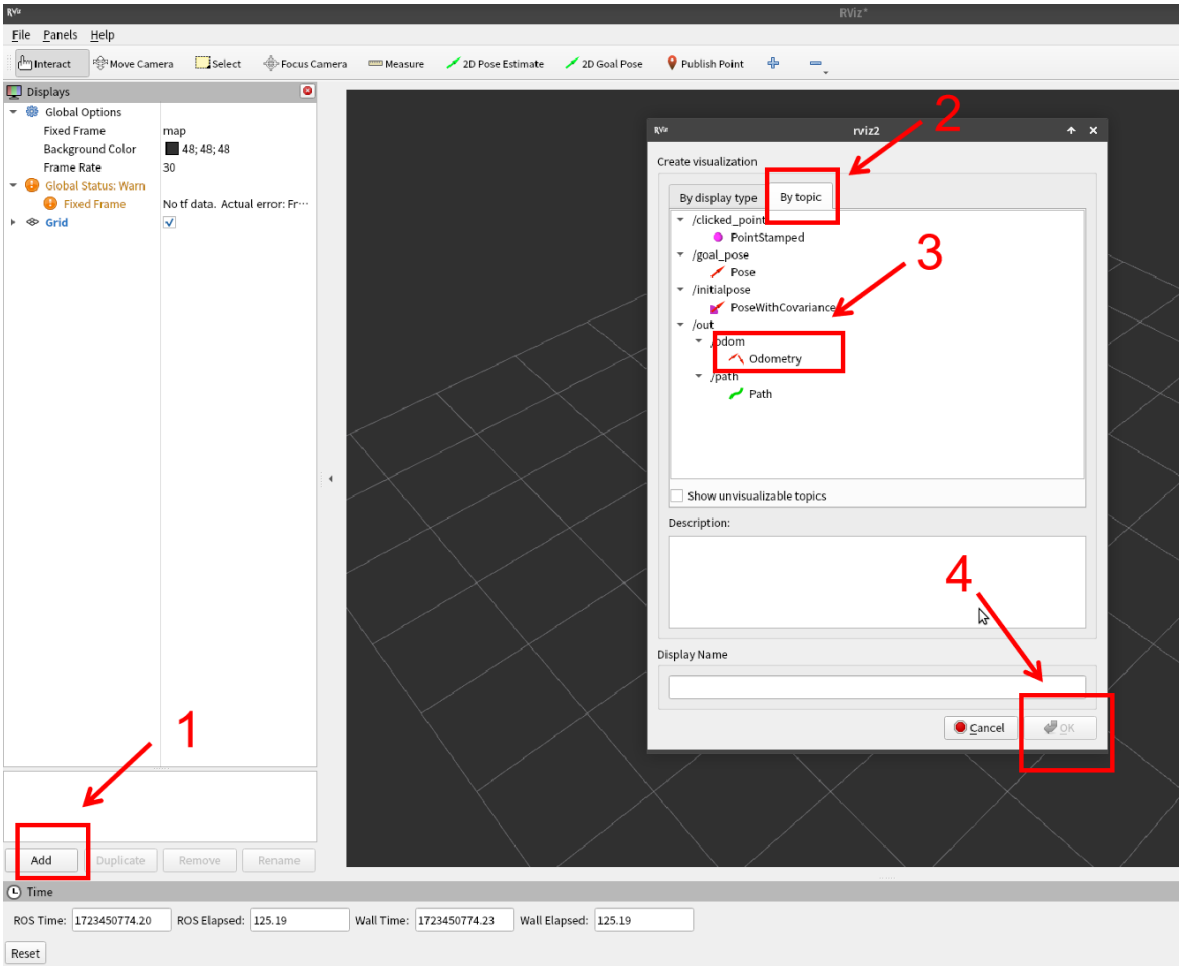
进入rviz可视化界面。

在bags文件夹下打开终端，输入

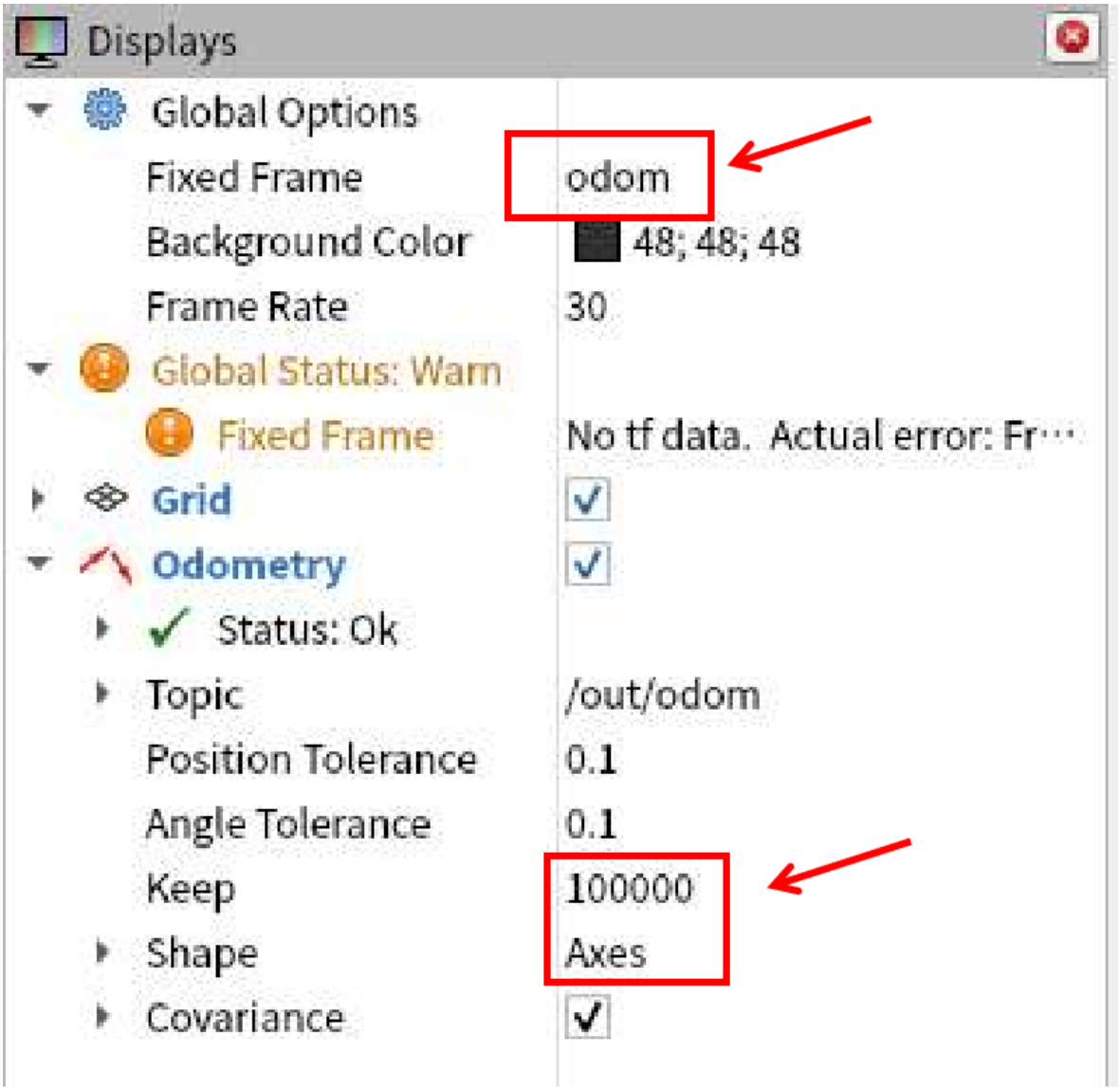
```
ros2 bag play navigation01/
```

将navigation01包中的消息载入到rviz中。

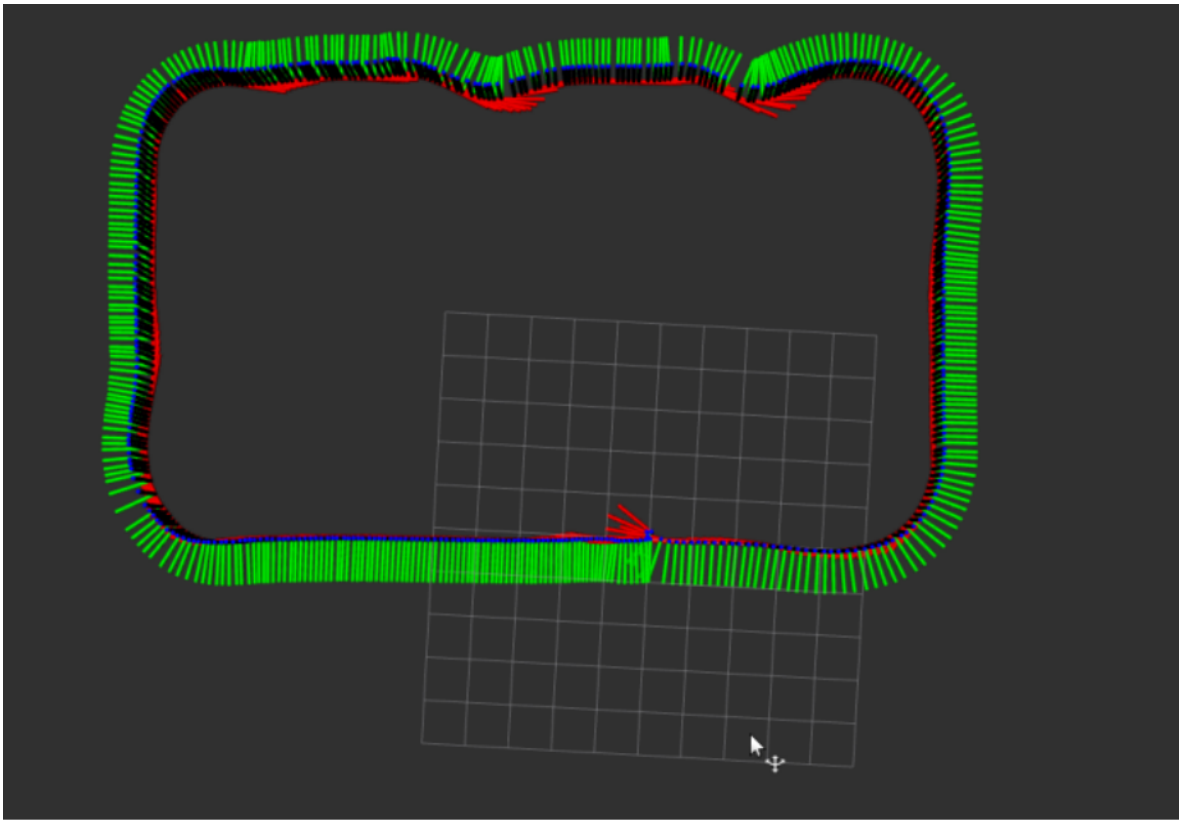
在rviz界面中，点击Add——By topic—— 选择里程计数据话题 —— 点击OK。



将rviz中的Fixed_Frame参数修改为odom。将shape参数修改为Axes显示坐标系。将keep参数修改为100000显示机器人运动的整个轨迹。



等待一段时间后，可以显示机器人导航运动的完整轨迹。这里显示的是在避障模式下机器人的轨迹，从图中可以看到机器人的轨迹出现一定的弯曲，表示在该位置机器人绕过障碍物并返回原路线。



接着查看导航实验数据的可视化结果。打开终端，输入

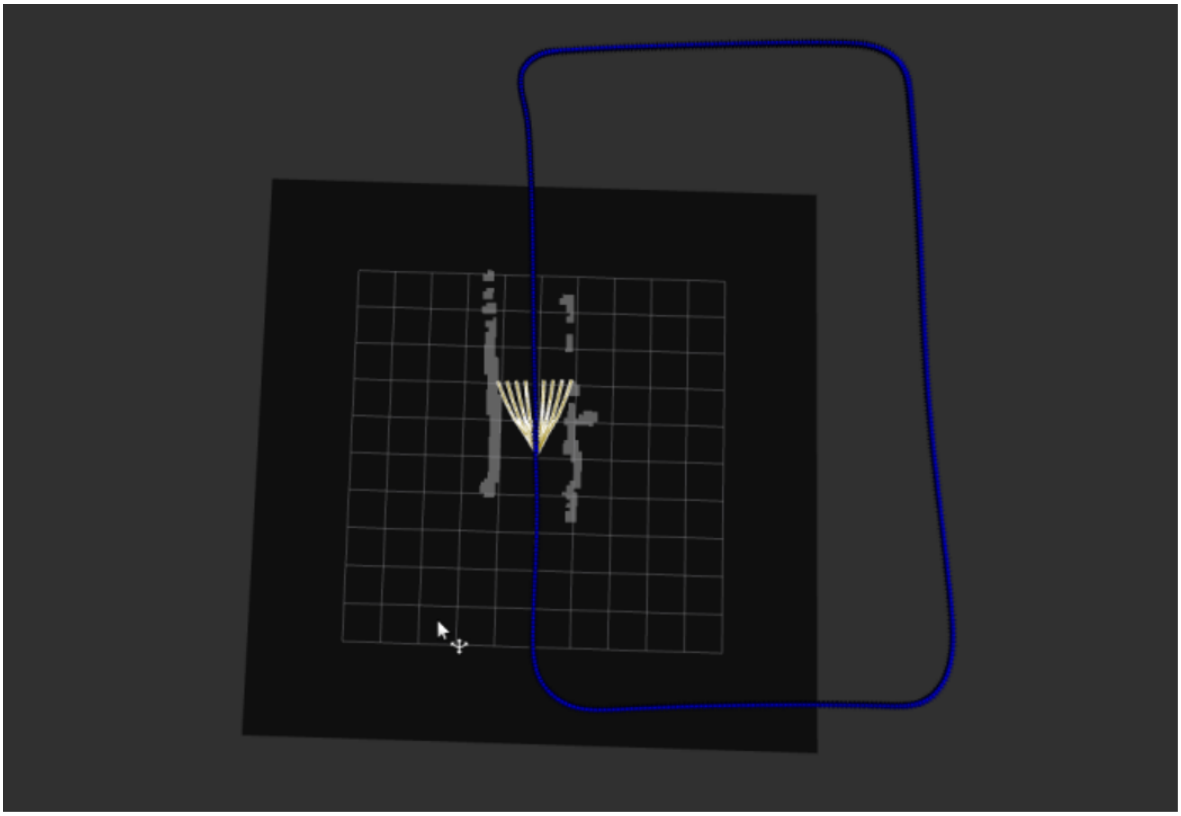
```
./run_navi_replay.sh
```

运行脚本，rviz窗口会自动弹出。

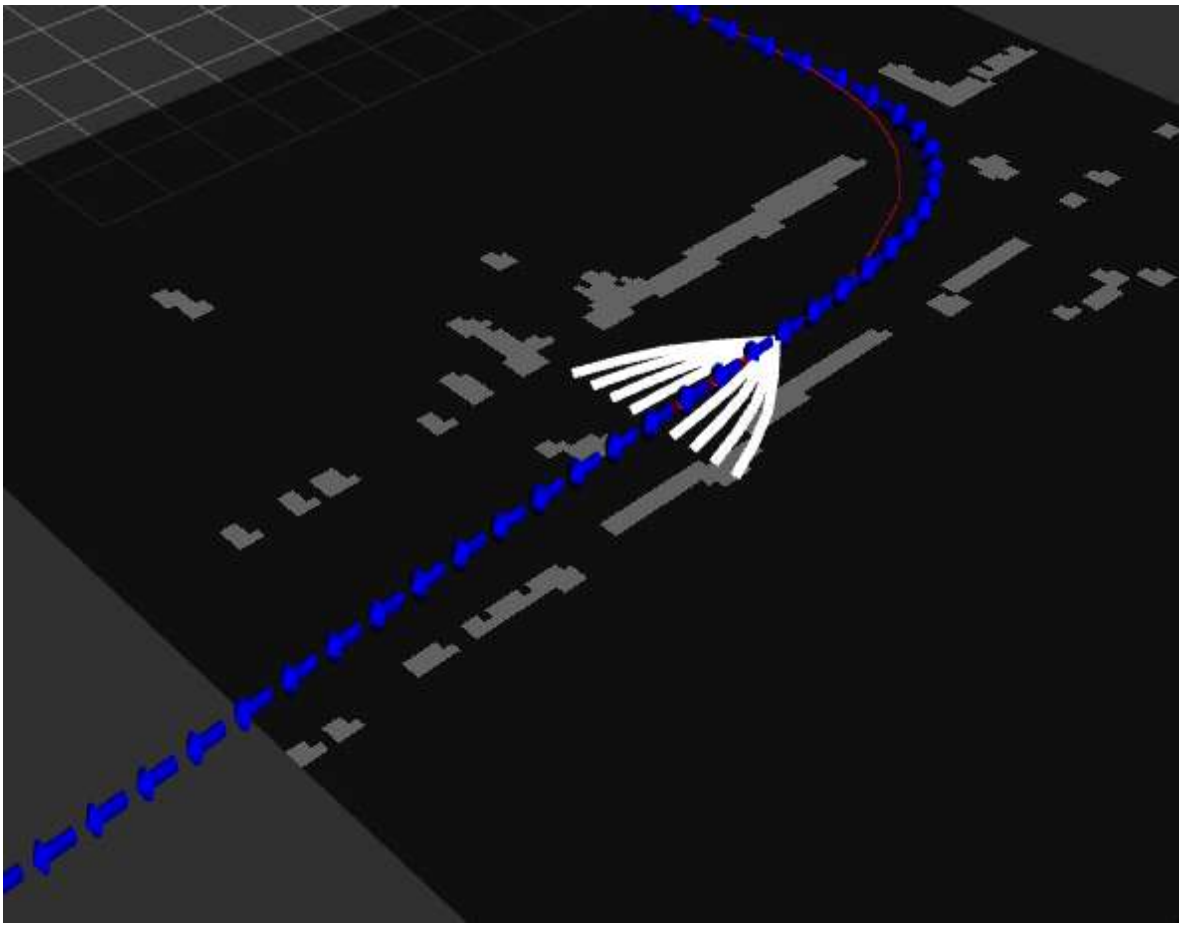
进入navigation文件夹，进入data文件夹下的bags文件夹，在bags文件夹下打开一个新的终端，输入

```
ros2 bag play navigation02/
```

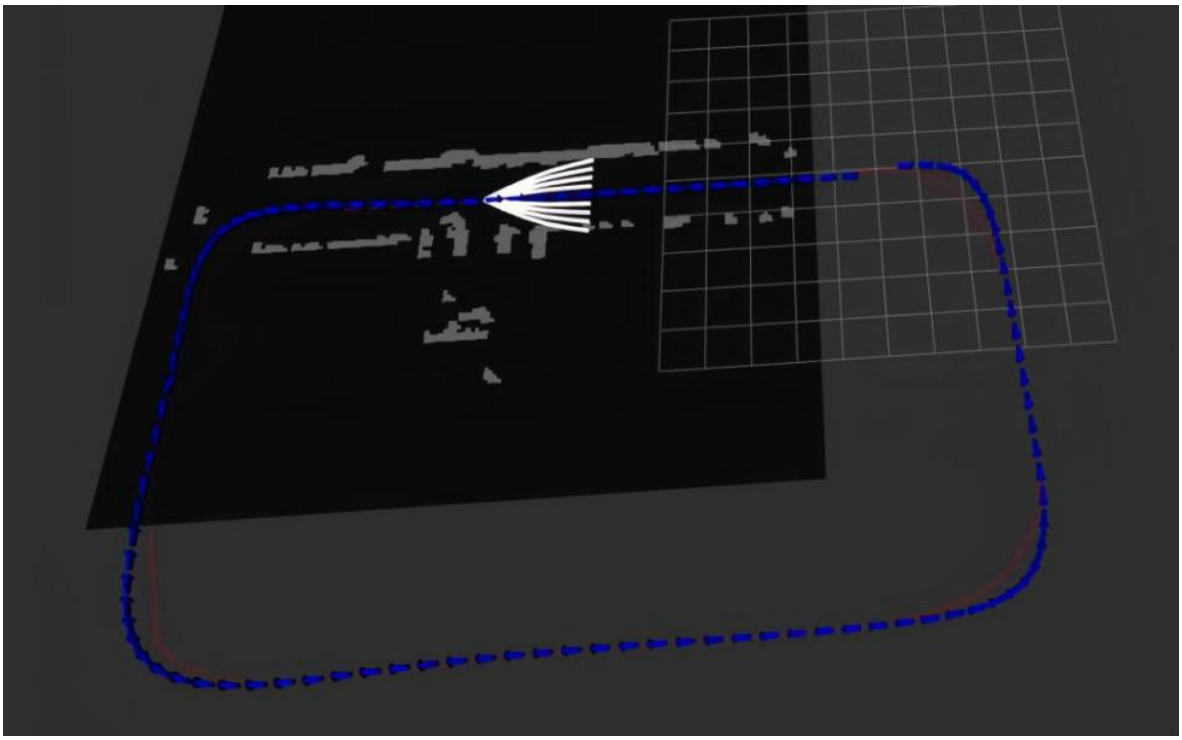
将navigation02包中的消息载入到rviz中。这时可以看到rviz界面中出现一条蓝色的线，这条线代表建图过程录入的机器人的轨迹。



机器人前方的白线代表机器人可以选择的运动轨迹，当前方蓝色的轨迹上出现障碍物，机器人可以选择白线的轨迹进行避障。



图中的红线代表机器人的实际运动轨迹，由可视化结果可以看出，机器人的实际运动轨迹与设定的轨迹基本相符。



6.2 实验小结

参考文献

[1] ArduPilot官网，https://ardupilot.org/
[2] ArduPilot开源资料，https://github.com/ArduPilot/ardupilot
[3] Ubuntu操作系统官网，https://cn.ubuntu.com/
[4] ROS官网，https://www.ros.org/
[5] Foxglove官网，https://foxglove.dev/